APRIL 13, 2023

ASSIGNMENT 2 REPORT

SUBHOJEET ROY & MEHUL MITTAL
subhojee & mehulmit

UBIT	PART	Contribution Percentage
mehulmit	1 ,2, 3 &4	50%
subhojee	1 ,2, 3 &4	50%

Part 1

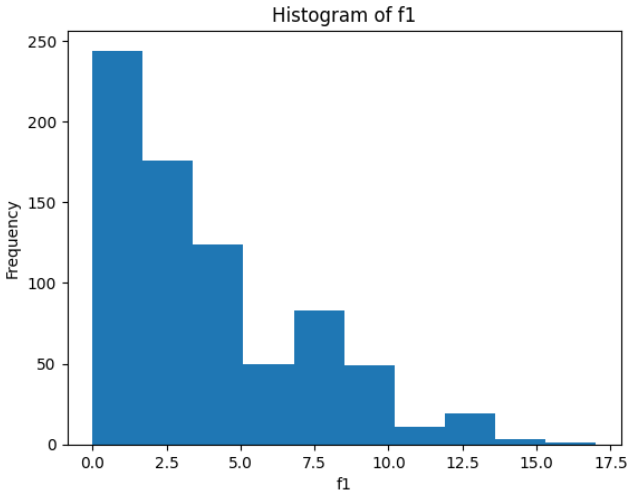
1. The Dataset consists of 8 features including the Target feature. The shape of the dataset is (760,8).

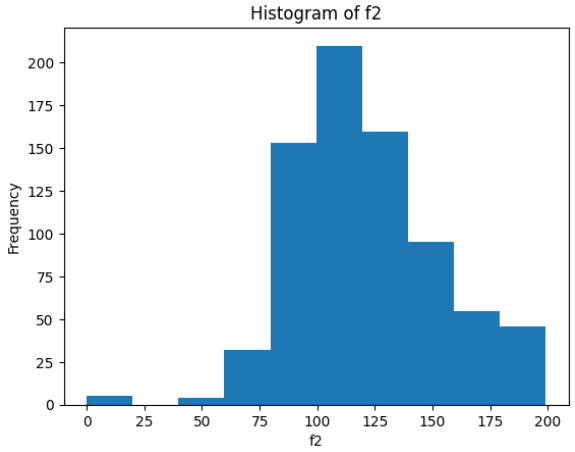
```
 f1          int64
f2          int64
f3          Int64
f4          int64
f5          int64
f6        float64
f7        float64
target      Int64
dtype: object

Shape of Data:  (760, 8)
```

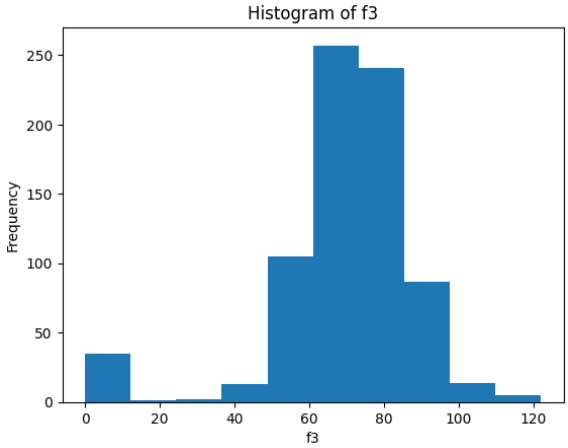
	f1	f2	f3	f4	f5	f6	f7	target
count	760.000000	760.000000	760.000000	760.000000	760.000000	760.000000	760.000000	760.000000
mean	3.834211	120.969737	69.119737	20.507895	80.234211	31.998684	0.473250	0.350000
std	3.364762	32.023301	19.446088	15.958029	115.581444	7.899724	0.332277	0.477284
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	0.000000
25%	1.000000	99.000000	63.500000	0.000000	0.000000	27.300000	0.243750	0.000000
50%	3.000000	117.000000	72.000000	23.000000	36.000000	32.000000	0.375500	0.000000
75%	6.000000	141.000000	80.000000	32.000000	128.250000	36.600000	0.627500	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	1.000000

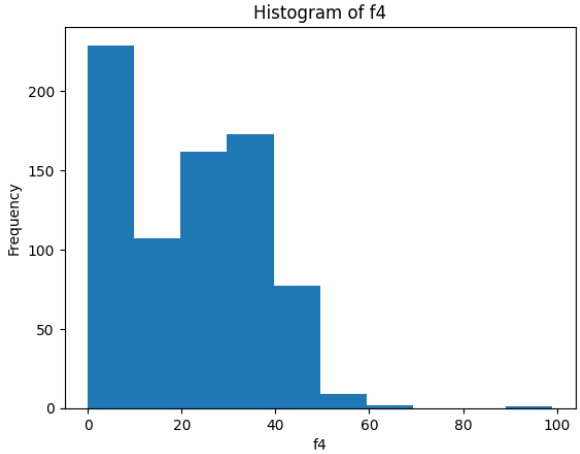
The various datatypes of all the features, the shape, and the statistics of the dataset.

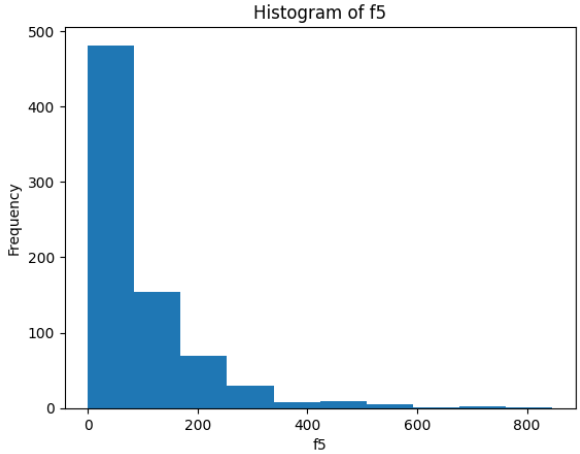
2. a) Histogram Plots: Based on the dataset we have 7 classes, the Hist plot gave an indication of how skewed each of the features are. We observed f2, f3 and f6 are almost normally distributed. For the other features the density of values were closer towards 0.

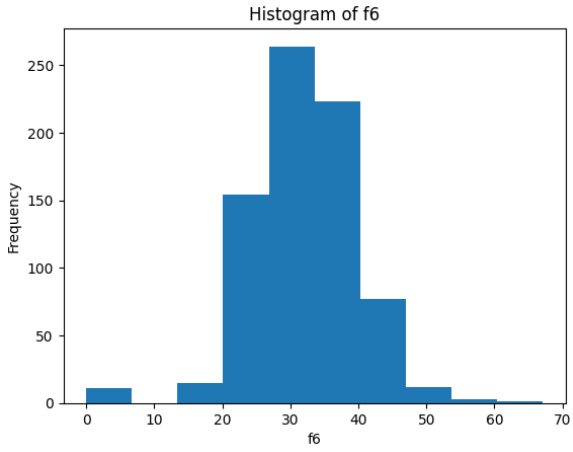


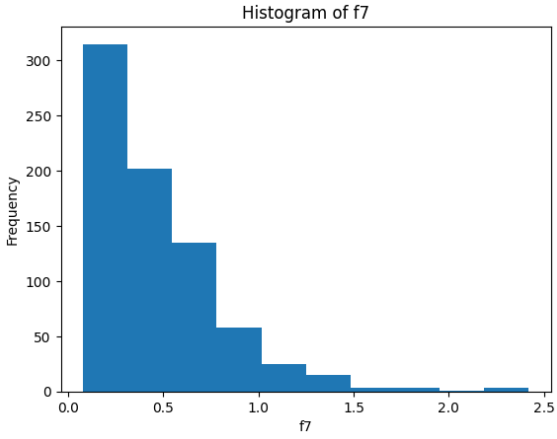




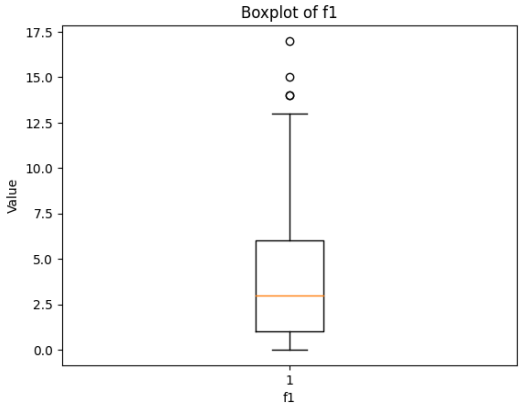


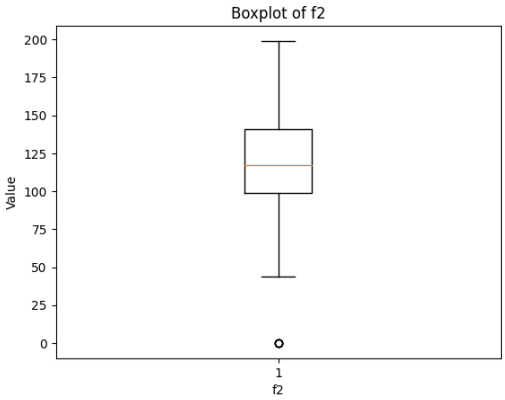


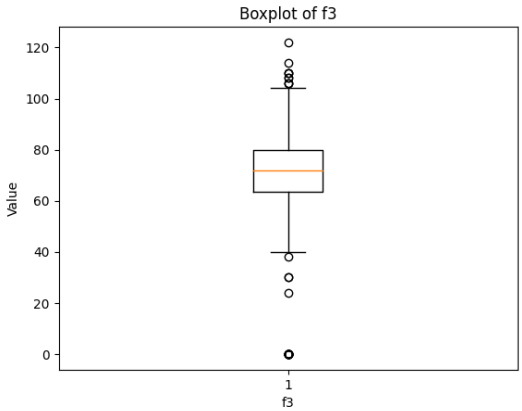


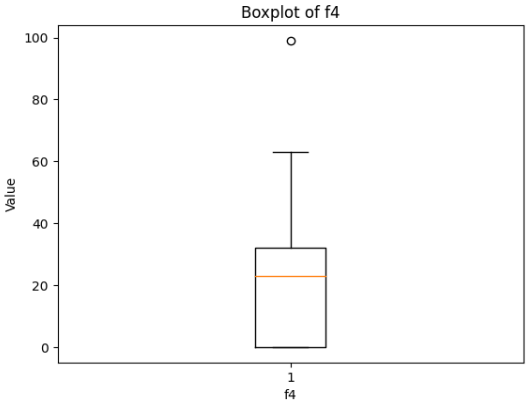


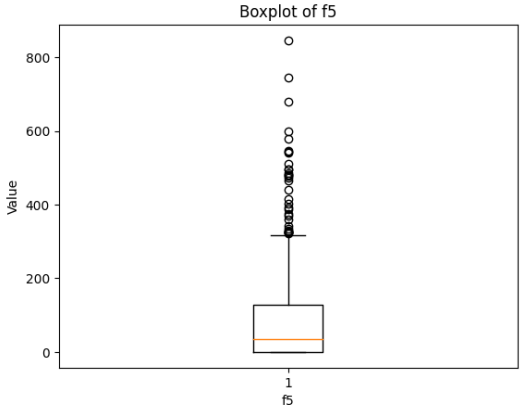
b) <u>Box Plots:</u> Box plots for the features indicated that the f3, f5, f6 and f7 features contained many outliers which need to be handled before we model.

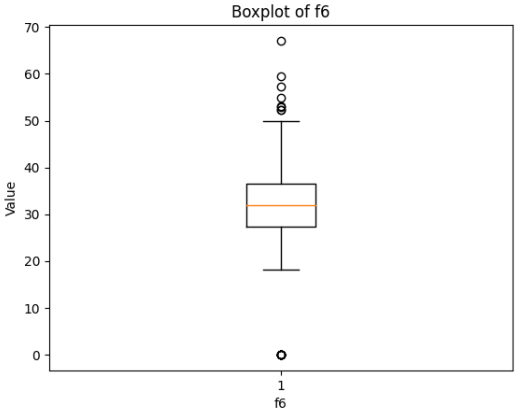

Boxplot of f1
Value
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1
f1

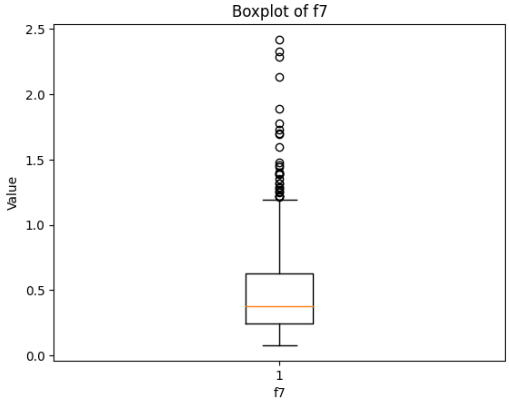

Boxplot of f2
Value
200
175
150
125
100
75
50
25
0
1
f2


Boxplot of f3
Value
120
100
80
60
40
20
0
1
f3

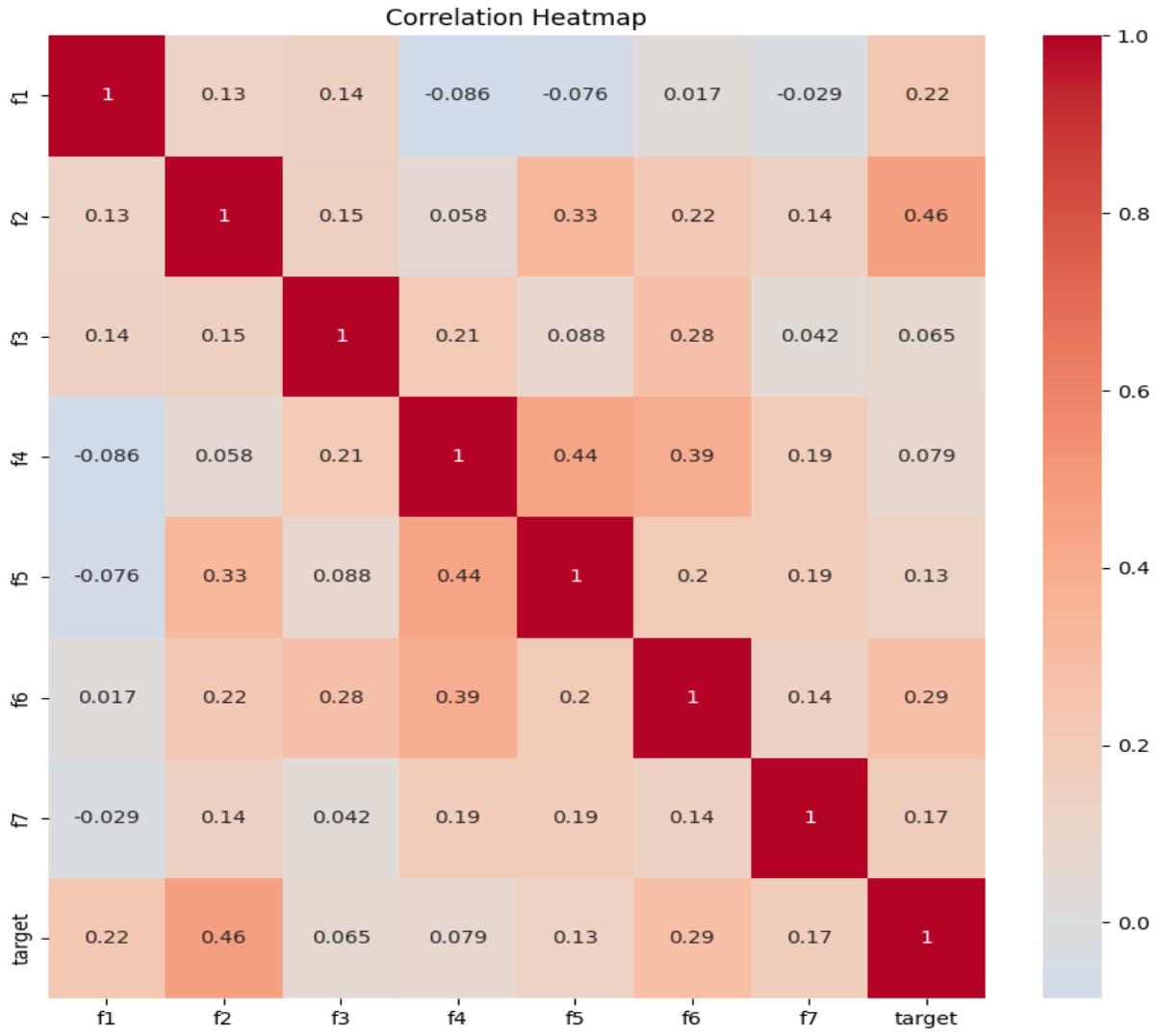

Boxplot of f4
Value
100
80
60
40
20
0
1
f4


Boxplot of f5
Value
800
600
400
200
0
1
f5


Boxplot of f6
Value
70
60
50
40
30
20
10
0
1
f6


Boxplot of f7
Value
2.5
2.0
1.5
1.0
0.5
0.0
1
f7

c) Correlation Matrix between Features: Features f2 and f6 seem to be highly correlated w.r.t the target while f4 and f5, and f6 and f4 are highly correlated with each other.

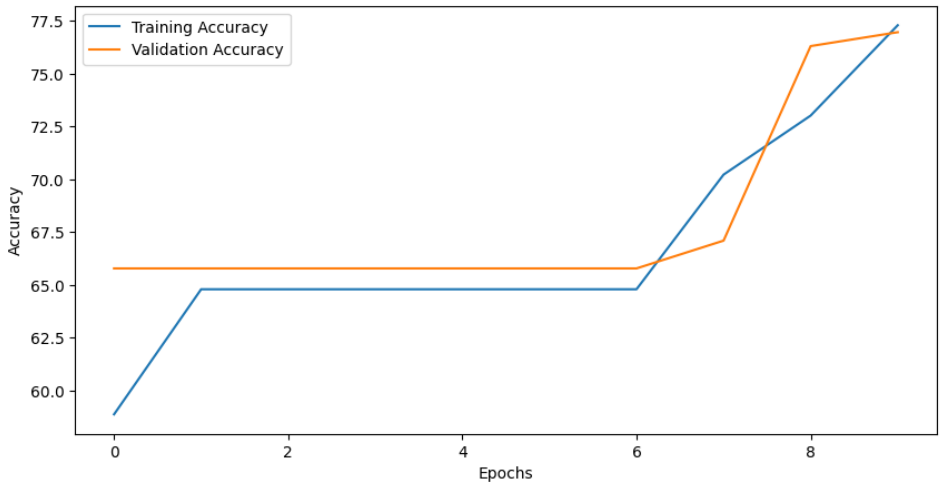


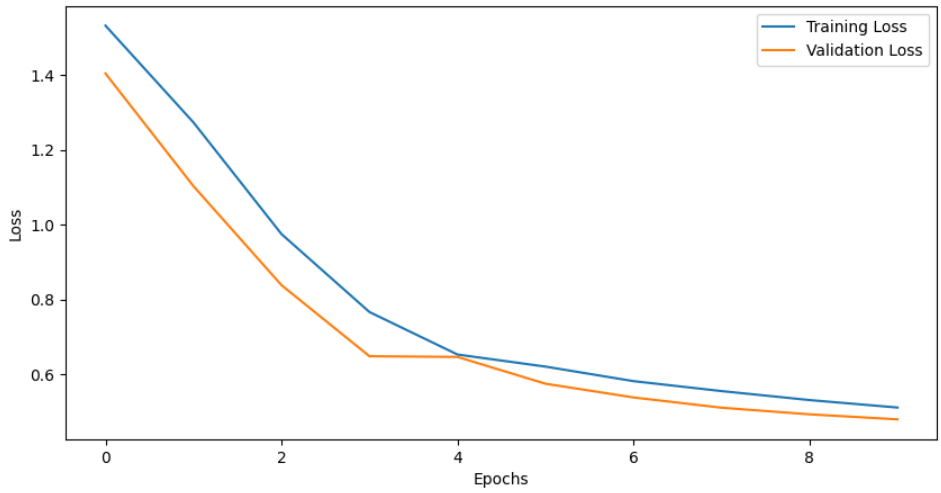
3. Initially accuracy came to around 65%. We used RobustScaler from sklearn to preprocess our input features to reduce the effect of outliers in the dataset. We also dropped features f3 and f4 as these features had very low correlation w.r.t our target feature.

4. This was the Neural Network architecture we used.

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(nn.Linear(5*1, 512),
                                               nn.ReLU(),
                                               nn.Linear(512, 512),
                                               nn.ReLU(),
                                               nn.Linear(512, 512),
                                               nn.ReLU(),
                                               nn.Linear(512, 512),
                                               nn.ReLU(),
                                               nn.Linear(512, 512),
                                               nn.ReLU(),
                                               nn.Linear(512, 512),
                                               nn.ReLU(),
                                               nn.Linear(512, 5),
                                               )
```

5.





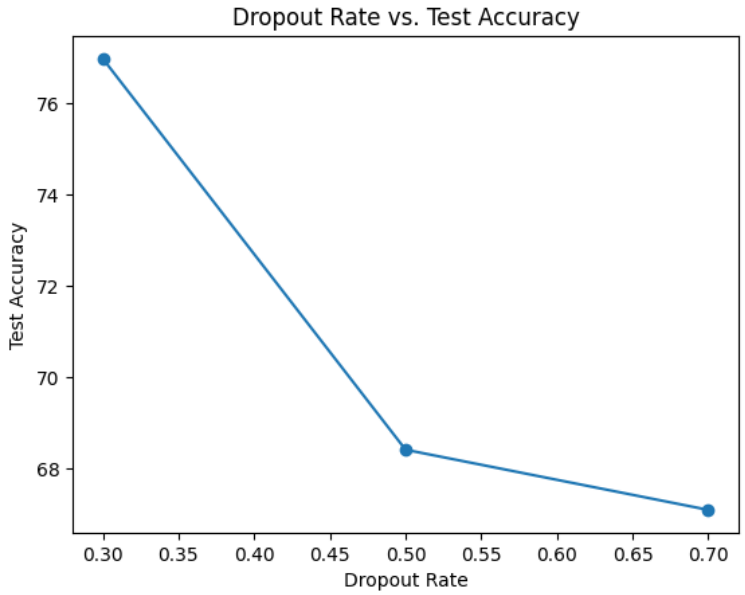
Part 2:

1.

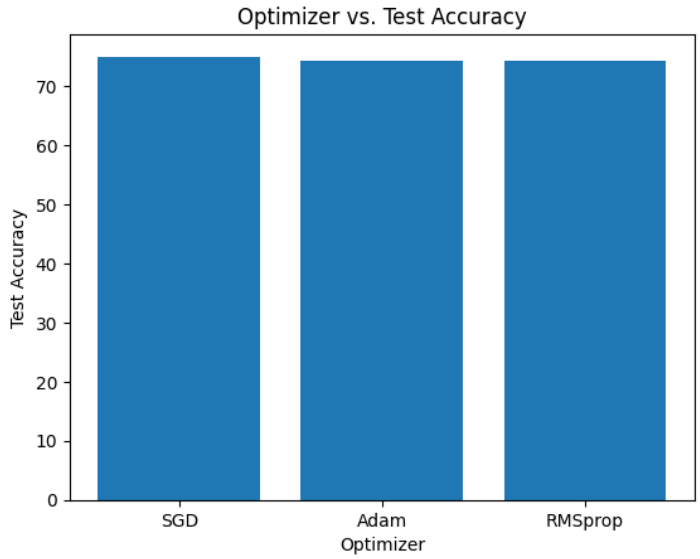
	Setup 1	Test Accuracy(%)	Setup 2	Test Accuracy	Setup 3	Test Accuracy(%)
Dropout	0.3	76.97	0.5	68.11	0.8	65.79
Optimizer	SGD	75	ADAM	74.34	RMSProp	74.34
Activation Function	RELU	76.97	Sigmoid	65.79	Leaky RELU	75

Best Combination Found was with Dropout = 0.3 , Optimizer as SGD and Activation Function as RELU. Used this Setup as Base Model for doing Step 2.

2.

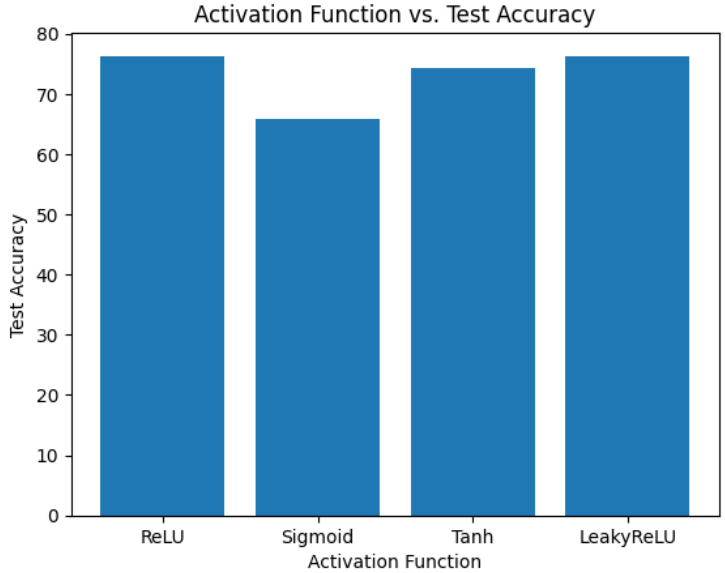


As we started increasing the Dropout Rate the Accuracy started falling, maybe our model is not able to learn about the essential features of our data.



SGD performed slightly better than Adam optimizer and RMSProp with our dataset. With the following result for each of them:

Test Accuracy with optimizer SGD	75%
Test Accuracy with optimizer Adam	74.34%
Test Accuracy with optimizer RMSprop	74.34%

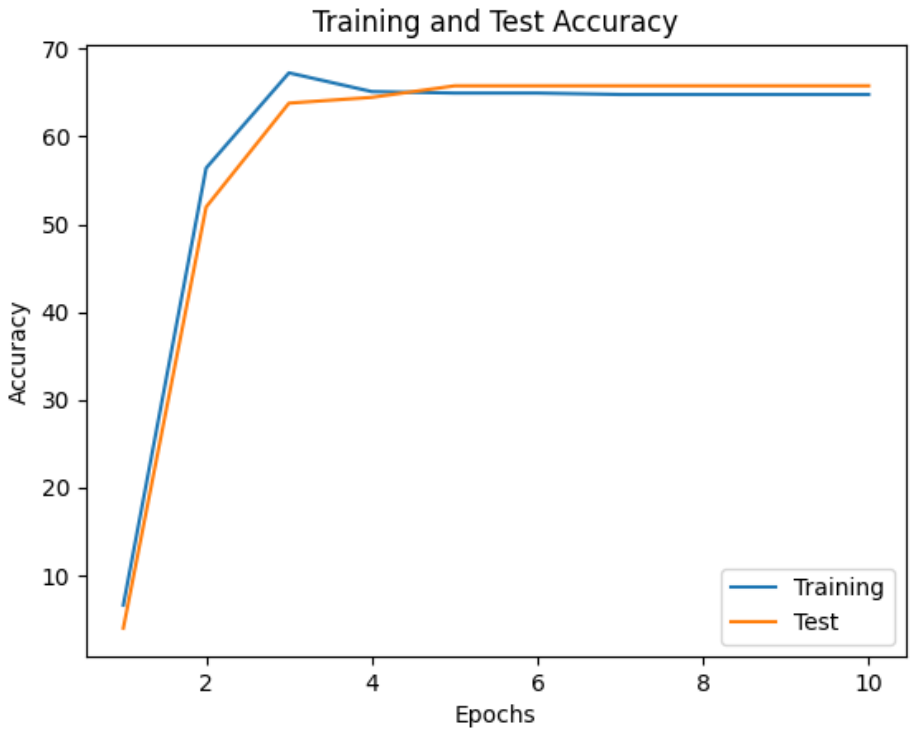


RELU and Leaky RELU as activation functions performed at par with each other while accuracy with tanh and sigmoid were a little lower.

Test Accuracy with activation function ReLU	76.97%
Test Accuracy with activation function Sigmoid	65.79%
Test Accuracy with activation function Tanh	75 %
Test Accuracy with activation function LeakyReLU	76.32%

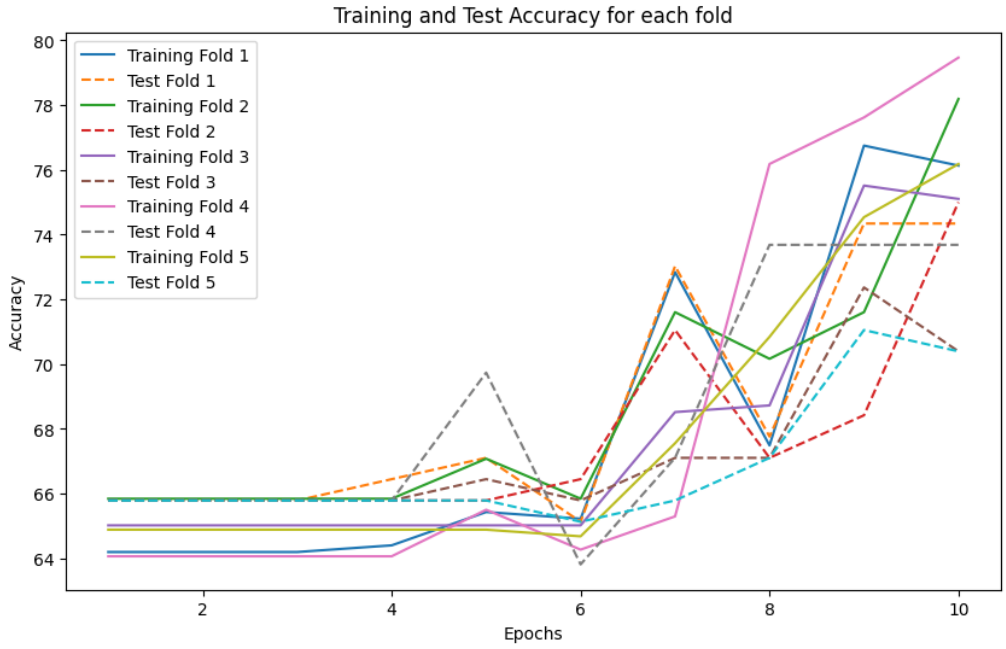
3. For the Base Model we tried the following setup with the parameters we got as part of doing Step 1, we used Dropouts as 0.3 , RELU activation and SGD :

```
class NeuralNetworkBase(nn.Module):
    def __init__(self):
        super(NeuralNetworkBase, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(nn.Linear(5*1, 512),
                                               nn.ReLU(),
                                               nn.Dropout(0.3),
                                               nn.Linear(512, 512),
                                               nn.ReLU(),
                                               nn.Dropout(0.3),
                                               nn.Linear(512, 512),
                                               nn.ReLU(),
                                               nn.Dropout(0.3),
                                               nn.Linear(512, 512),
                                               nn.ReLU(),
                                               nn.Dropout(0.3),
                                               nn.Linear(512, 512),
                                               nn.ReLU(),
                                               nn.Dropout(0.3),
                                               nn.Linear(512, 512),
                                               nn.ReLU(),
                                               nn.Dropout(0.3),
                                               nn.Linear(512, 5),
                                               )

```

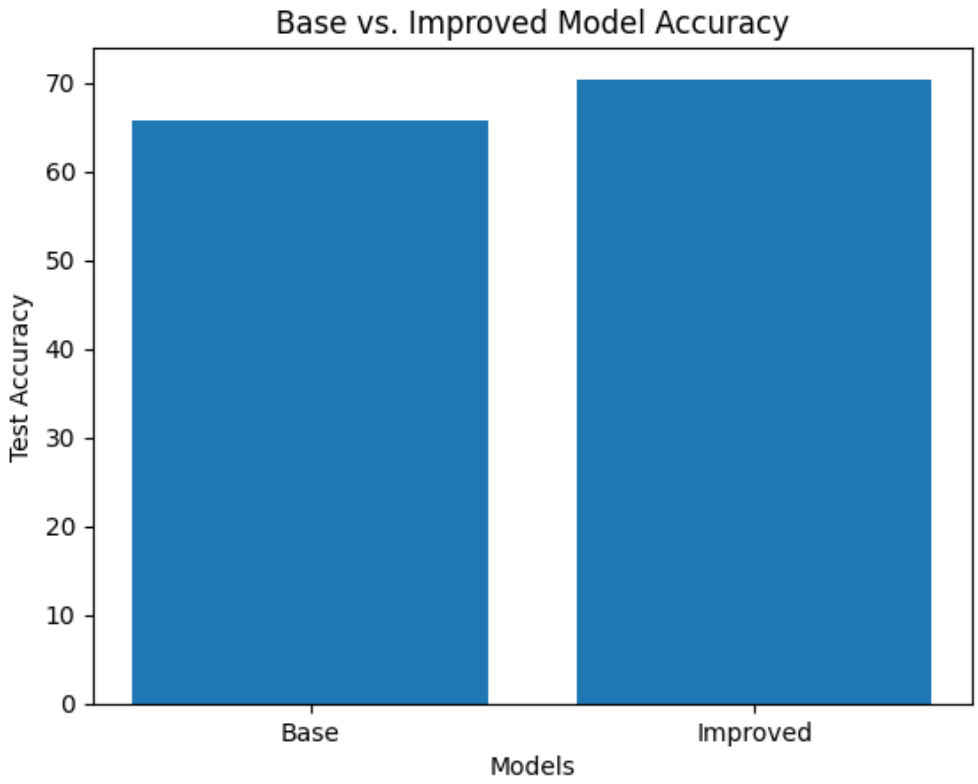


We Optimized Neural Network Base class and implemented using early stopping, k-folds cross-validation with 5 folds, learning rate scheduler, and weight initialization using Xavier initialization.

To try and improve the performance of our model.



Final Model results



As we can see above, the improved model is considerably better, showing us how using these optimization techniques can help us improve the performance of the model.